Επεξεργασία και Διαχείριση Δεδομένων σε Δίκτυα Αισθητήρων

Αναφορά Πρώτου Μέρους.

Καπενεκάκης Πέτρος 2018030034
Πιπεράκης Γιώργος 2018030012

Κομμάτια Υλοποίησης Από Κάθε Φοιτητή:

Το σύνολο της εργασία πραγματοποιήθηκε και από τους δύο φοιτητές, τόσο το development όσο και το testing του προγράμματος μας.

Βοηθητικό Πρόγραμμα:
Topology.py

Η γλώσσα προγραμματισμού το προγράμματος που επιλέξαμε είναι η python.

Το παρεχόμενο σκριπτ Python δημιουργεί μια τοπολογία δικτύου για ένα πλέγμα κόμβων, αποθηκεύοντας τη σε ένα αρχείο κειμένου κατάλληλο για προσομοιώσεις. Το πλέγμα έχει διαστάσεις D×D, όπου το D είναι μια τιμή που καθορίζεται από τον χρήστη, και κάθε κόμβος αναπαρίσταται με μια μοναδική τιμή βασισμένη στη θέση του στο πλέγμα.

Το πρόγραμμα υπολογίζει την Ευκλείδεια απόσταση μεταξύ κάθε ζεύγους κόμβων για να καθορίσει αν βρίσκονται εντός μιας συγκεκριμένης ακτίνας επικοινωνίας η οποία επίσης ορίζεται από τον χρήστη. Αν οι κόμβοι βρίσκονται εντός αυτής της ακτίνας, δημιουργείται μια αμφίδρομη σύνδεση μεταξύ τους. Οι συνδέσεις αυτές αποθηκεύονται και γράφονται σε ένα αρχείο με τη μορφή των αρχείων topology που μας παρέχονται στον βοηθητικό κώδικα.

Πρόγραμμα 1:

Για την υλοποίηση της άσκησης χρειάστηκε να δημιουργηθούν καινούργια αρχεία αλλά και να τροποποιηθούν ήδη υπάρχοντα αρχεία που μας δόθηκαν. Οι αλλαγές αυτές, καθώς και τα αρχεία που δημιουργήθηκαν παρουσιάζονται αναλυτικά παρακάτω.

Δημιουργήθηκε το SensorValue.h

Αποτελεί αρχείο header, το οποίο ορίζει μία καινούργια δομή δεδομένων με ονομασία sensor_value η οποία περιλαμβάνει τις παρακάτω τιμές, οι οποίες είναι οι τιμές που μεταδίδουμε μεταξύ των κόμβων του δέντρου μας.

Οι τιμές είναι οι εξής:

SensorValue: Η τιμή του αθροίσματος που στέλνουμε
MaxSensorValue: Η μέγιστη τιμή που έχει συναντήσει ο αισθητήρας
Count: Πόσοι αισθητήρες έχουν στείλει τιμή

Δημιουργήθηκαν τα αρχεία AggregationC και Aggregator:

Το αρχείο AggregationC αποτελεί αρχείο Module το οποίο περιέχει όλες τις συναρτήσεις που δημιουργήθηκαν για τον υπολογισμό των συναθροιστικών ερωτημάτων.

Το αρχείο Aggregator αποτελεί το interface του Module AggregationC.

Οι συναρτήσεις που περιέχονται στο αρχείο AggregationC είναι οι εξής:

readSeedFromFile():
Αποτελεί βοηθητική συνάρτηση η οποία διαβάζει έναν τυχαίο αριθμό από το αρχείο seed.txt. Ο αριθμός αυτός χρησιμοποιείται ως seed για την παραγωγή διαφορετικών τυχαίων μεταβλητών στην συνάρτηση ChooseAggregation().

Command ChooseAggregation:
Δημιουργεί έναν ψευδοτυχαίο αριθμό έτσι ώστε να επιλεγεί η συναθροιστική συνάρτηση. Για να εξασφαλίσουμε διαφορετικό αριθμό σε κάθε εκτέλεση χρησιμοποιούμε έναν τυχαίο αριθμό seed ο οποίος παράγεται απο το mySimulation.py script και αποθηκεύεται στο αρχείο seed.txt. Για τον αριθμό 1 επιλέγουμε τη συνάρτηση max και για τον αριθμό 2 την average.

Command finalizeAggregation:
Αρχικά ενημερώνει τις τιμές του κόμβου 0 και στη συνέχεια διαλέγει την κατάλληλη συναθροιστική συνάρτηση.

Command finalizeAggregationOptional:
Συνδυάζει τις δύο παραπάνω συναρτήσεις σε μία. Χρησιμοποιήθηκε σε περίπτωση που θέλουμε ο κόμβος 0 να διαλέγει άλλη συναθροιστική συνάρτηση κάθε εποχή.

Command finishSendingData:
Αποτελεί Task υπεύθυνο για την αποστολή δεδομένων στον πατέρα του κόμβου. Η συνάρτηση εξασφαλίζει ότι στέλνονται μόνο έγκυρα δεδομένα και ότι δεν θα έχουμε παρά μόνο μια αποστολή την ίδια χρονική στιγμή.

command sendAggregatedData:
Ενημερώνει τις κατάλληλες τιμές του αισθητήρα και καλεί το task finishSendingData για την αποστολή δεδομένων.

event AMsend.sendDone:
Αποτελεί event που καλείται όταν η αποστολή του μηνύματος ολοκληρωθεί. Αν το μήνυμα στάλθηκε με επιτυχία επαναφέρει τις τιμές του αισθητήρα για την επόμενη εποχή. Εναλλακτικά καλεί ξανα το task finishSendingData με σκοπό την σωστή αποστολή δεδομένων.

event Receive.receive:

Αποτελεί event που καλείται όταν ο κόμβος λάβει δεδομένα. Είναι υπεύθυνο για να καλέσει τη συνάρτηση συλλογής δεδομένων.

command CollectData:

Συλλέγει τα δεδομένα που ο αισθητήρας έλαβε από τους άλλους αισθητήρες και ενημερώνει τις κατάλληλες τιμες (όπως για παράδειγμα τη μέγιστη τιμή).

command aggregateMax:

Η συναρτιστική συνάρτηση που μας ζητήθηκε. Τυπώνει τη μέγιστη τιμή στην κονσόλα.

command aggregateAvg:

Η συναρτιστική συνάρτηση που μας ζητήθηκε. Εφόσον ο αισθητήρας έχει λάβει τιμές από άλλους αισθητήρες υπολογίζει τον μέσο όρο αυτών και τον τυπώνει στην κονσόλα.

command generateRandomSensorValue:

Παράγει μια τυχαία ακέραια τιμή στο διάστημα 1 έως 50.

command initialGenerateRandomSensorValue:

Παράγει μια τυχαία ακέραια τιμή από το 1 έως το 50, φροντίζοντας η τιμή αυτή να μην απέχει παραπάνω από 30% από την προηγούμενη μέτρηση του αισθητήρα.

Τροποποιήθηκε το αρχείο SRTreeAppC:

Έγινε η διασύνδεση των παρακάτω Components:

AggregationC: καθώς είναι καινούργιο module
RandomC: καθώς γίνεται χρήση του από το AggregationC
AggregationTimerC: timer που εξασφαλίζει εποχή 40s
DepthDelayTimerC: Timer που εξασφαλίζει τον συγχρονισμό μετάδοσης.
AggregationSenderC: AMSender για την μετάδοση των δεδομένων
AggregationReceiverrC: AMReceiver για την παραλαβή δεδομένων

Τροποποιήθηκε το αρχείο SRTreeC:

Δημιουργήθηκαν οι τροποποιήθηκαν οι παρακάτω συναρτήσεις.

event void Boot.booted():

Αρχικά Κατά τη διάρκεια του Event Boot,booted() ξεκινάμε τον AggregationTimer, ο οποίος τρέχει επαναλαμβανόμενα, με διάρκεια 40960ms φού στην πραγματικότητα ένα δευτερόλεπτο αποτελείται από 1024ms. Με αυτόν τον τρόπο δημιουργούμε έναν timer που χτυπάει κάθε 40 δευτερόλεπτα.

event void AggregationTimer.fired()

Μόλις κληθεί ο AggregationTimer έχουμε δημιουργήσει το event AggregationTimer.fired().
Εκεί αρχικά οι κόμβοι οι οποίοι δεν βρίσκονται στο δέντρο μας εκείνη την εποχή σταματούν τη λειτουργία τους αφού δεν μας απασχολούν τα δεδομένα τους, με σκοπό την όσο δυνατόν μικρότερη λειτουργία τους

Στη συνέχεια αν βρισκόμαστε στη πρώτη εποχή οι κόμβοι παράγουν μια τυχαία τιμή στο διάστημα 1 έως 50. Οποιαδήποτε άλλη εποχή παράγουν μια τυχαία τιμή η οποία όμως δεν απέχει περισσότερο από 30% σε σχέση με την προηγούμενη.

Έπειτα καλείται ο DepthDelayTimer με μία καθυστέρηση. Η καθυστέρηση αυτή υπολογίζεται με βάση το βάθος του κόμβου στο δέντρο. Όσο πιο βαθιά στο δέντρο τόσο μικρότερη καθυστέρηση με σκοπό να πετύχουμε τον τρόπο μετάδοσης του Tag.
Οι κόμβοι με ίδιο βάθος μεταδίδουν σε λίγο διαφορετικό χρόνο με σκοπό να μειώσουμε τις συγκρούσεις το μηνυμάτων για την ομαλότερη λειτουργία του προγράμματος.
Ο μόνος μας περιορισμός είναι ότι έχουμε υποθέσει μέγιστο βαθος δέντρο ίσο με το 30. Σε περίπτωση που το υπερβούμε αυτό υπάρχει περίπτωσή οι κόμβοι με βάθος μεγαλύτερο του 30 να μην μεταδίδουν μηνύματα με σωστό συγχρονισμό.

event void DepthDelayTimer.fired()

Ο κόμβος ελέγχει αν έχει στείλει δεδομένα αυτή την εποχή, ώστε να εξασφαλίσουμε μία αποστολή δεδομένων ανά εποχή ανά κόμβο.
Ακόμα ελέγχει αν έχει λάβει δεδομένα από όλα τα παιδιά του και δεν μεταδίδει μέχρι να λάβει από όλα
Εφόσον έλαβε από όλα αν είναι ο σταθμός βάσης υπολογίζει την συναθροιστική συνάρτηση με βάση τα δεδομένα που έχει λάβει καθώς και τα δικά του
Εναλλακτικά αν είναι κάποιος άλλος κόμβος στέλνει τα δεδομένα στον πατέρα του.

event message_t* RoutingReceive.receive

αυξάνουμε την μεταβλητή receivedFromChildren αφού λάβαμε μήνυμα από παιδί

task void receiveNotifyTask

Ελέγχουμε αν ο αποστολέας είναι παιδί μας και αν είναι αυξάνουμε το childCount. Αν δεν είναι το μειώνουμε

Τροποποιήθηκε το αρχείο mySimulation.py:

Προσθέσαμε τα παρακάτω κανάλια για το Debugging του κώδικα. Παραπάνω πληροφορίες για το πως χρησιμοποιούνται ακολουθούν στο επόμενο μέρος της αναφοράς.

```
t.addChannel("SensorValues",f)
t.addChannel("CustomSend",f)
t.addChannel("CustomReceive",f)
t.addChannel("CustomAggregationFunction",f)
t.addChannel("CustomDataChosen",f)
t.addChannel("Custom",f)
```

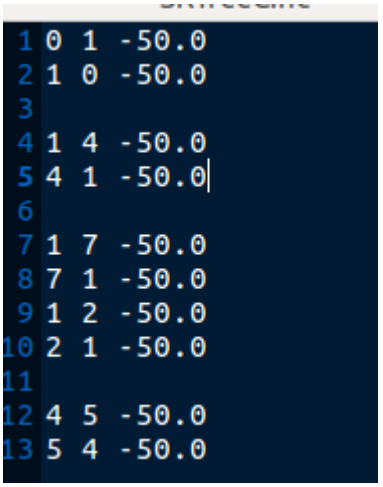
Ακόμα προσθέσαμε τις παρακάτω γραμμές:

```
import time #Pkapenekakis Gpiperakis

seed = int(time.time())  # Generate the seed
print("Saving RANDOM_SEED to file:", seed)

with open("seed.txt", "w") as f:
    f.write(str(seed))
```

Με αυτόν τον τρόπο δημιουργούμε ένα αρχείο με όνομα seed.txt. Που περιλαμβάνει ένα τυχαίο αριθμό. Ο αριθμός αυτός χρησιμοποιείται μετά στο Module AggregationC.

Παραδείγματα εκτέλεσης του προγράμματος:

Για όλα τα παρακάτω παραδείγματα χρησιμοποιούμε το παρακάτω αρχείο topology, παρόλα αυτά έχει επιβεβαιωθεί η λειτουργία του προγράμματος και για άλλα αρχεία :



Στα παρακάτω παραδείγματα βλέπουμε διαφορετικές συναθροιστικές συναρτήσεις ανά εποχή για να είναι ευκολότερη η επιβεβαίωση των στοιχείων μας σε κάποια παραδείγματα.
Στο παραδοτέο πρόγραμμα ο κόμβος 0 επιλέγει μια από τις δύο συναθροιστικές συναρτήσεις στην πρώτη εποχή και συνεχίζει να δουλεύει με αυτή καθόλη τη διάρκεια του simulation.

Για τον έλεγχο της δημιουργίας των τιμών έχουμε δημιουργήσει ένα καινούργιο κανάλι μέσα στο αρχείο mySimulation.py με όνομα SensorValues.
Παρακάτω φαίνονται οι τιμές που δημιουργήθηκαν για 3 εποχές.

```
  5   4   -50.0
0:0:50.000000010 DEBUG (0): Init Value generated for nodeID: 0 is: 8
0:0:50.000000010 DEBUG (1): Init Value generated for nodeID: 1 is: 15
0:0:50.000000010 DEBUG (2): Init Value generated for nodeID: 2 is: 22
0:0:50.000000010 DEBUG (4): Init Value generated for nodeID: 4 is: 50
0:0:50.000000010 DEBUG (5): Init Value generated for nodeID: 5 is: 7
0:0:50.000000010 DEBUG (7): Init Value generated for nodeID: 7 is: 35
AVG Aggregation Result: 22
0:1:30.000000010 DEBUG (0): Value generated for nodeID: 0 is: 9
0:1:30.000000010 DEBUG (1): Value generated for nodeID: 1 is: 14
0:1:30.000000010 DEBUG (2): Value generated for nodeID: 2 is: 18
0:1:30.000000010 DEBUG (4): Value generated for nodeID: 4 is: 41
0:1:30.000000010 DEBUG (5): Value generated for nodeID: 5 is: 5
0:1:30.000000010 DEBUG (7): Value generated for nodeID: 7 is: 28
MAX Aggregation Result: 41
0:2:10.000000010 DEBUG (0): Value generated for nodeID: 0 is: 9
0:2:10.000000010 DEBUG (1): Value generated for nodeID: 1 is: 18
0:2:10.000000010 DEBUG (2): Value generated for nodeID: 2 is: 14
0:2:10.000000010 DEBUG (4): Value generated for nodeID: 4 is: 35
0:2:10.000000010 DEBUG (5): Value generated for nodeID: 5 is: 5
0:2:10.000000010 DEBUG (7): Value generated for nodeID: 7 is: 20
AVG Aggregation Result: 16
```

Αρχικά στην πρώτη εποχή δημιουργούνται τυχαίες τιμές από το 1 έως 50 για κάθε κόμβο.
Στη συνέχεια παρατηρούμε ότι η τιμή που παράγεται για κάθε αισθητήρα δεν απέχει πάνω από 30% όπως μας ζητήθηκε.

Για τον έλεγχο της σωστής αποστολής μηνυμάτων με βάση το TAG, δηλαδή πρώτα στέλνουν δεδομένα τα παιδία και μετά οι γονείς, έχουμε δημιουργήσει ένα καινούργιο κανάλι μέσα στο αρχείο mySimulation.py με όνομα CustomSend.

```
  3   4   -30.0
0:1:1.994277933 DEBUG (5): Message sent successfully by node with id: 5 to the parent 4
0:1:2.425491304 DEBUG (7): Message sent successfully by node with id: 7 to the parent 1
0:1:2.436187738 DEBUG (4): Message sent successfully by node with id: 4 to the parent 1
0:1:2.444015507 DEBUG (2): Message sent successfully by node with id: 2 to the parent 1
0:1:2.889526365 DEBUG (1): Message sent successfully by node with id: 1 to the parent 0
AVG Aggregation Result: 22
0:1:41.989166269 DEBUG (5): Message sent successfully by node with id: 5 to the parent 4
0:1:42.419036874 DEBUG (7): Message sent successfully by node with id: 7 to the parent 1
0:1:42.436355583 DEBUG (4): Message sent successfully by node with id: 4 to the parent 1
0:1:42.447097765 DEBUG (2): Message sent successfully by node with id: 2 to the parent 1
0:1:42.887557992 DEBUG (1): Message sent successfully by node with id: 1 to the parent 0
MAX Aggregation Result: 41
0:2:21.998123126 DEBUG (5): Message sent successfully by node with id: 5 to the parent 4
0:2:22.422317494 DEBUG (7): Message sent successfully by node with id: 7 to the parent 1
0:2:22.442214924 DEBUG (4): Message sent successfully by node with id: 4 to the parent 1
0:2:22.451339683 DEBUG (2): Message sent successfully by node with id: 2 to the parent 1
0:2:22.893508885 DEBUG (1): Message sent successfully by node with id: 1 to the parent 0
AVG Aggregation Result: 16
0:3:1.996337858 DEBUG (5): Message sent successfully by node with id: 5 to the parent 4
0:3:2.419723515 DEBUG (7): Message sent successfully by node with id: 7 to the parent 1
0:3:2.437042225 DEBUG (4): Message sent successfully by node with id: 4 to the parent 1
0:3:2.450378385 DEBUG (2): Message sent successfully by node with id: 2 to the parent 1
0:3:2.888671878 DEBUG (1): Message sent successfully by node with id: 1 to the parent 0
MAX Aggregation Result: 44
```

Παρακάτω παρατηρούμε ότι αυτό όντως συμβαίνει καθώς: Πρώτα στέλνει δεδομένα ο κόμβος 5 στον κόμβο 4, αφού αποτελεί παιδί του. Στη συνέχεια στέλνουν δεδομένα οι κόμβοι 7, 4 και 2 στον κόμβο 1. Τέλος ο κόμβος 1 στέλνει στον κόμβο 0 (κόμβο βάσης) τα δεδομένα που έχει συλλέξει.

Για τον έλεγχο της σωστής λήψης μηνυμάτων έχουμε δημιουργήσει ένα καινούργιο κανάλι μέσα στο αρχείο mySimulation.py με όνομα CustomReceive.

```
9:1:1.995407077 DEBUG (4): Node 4 received data
9:1:1.995574923 DEBUG (5): Message sent successfully by node with id: 5 to the parent 4
9:1:2.427597004 DEBUG (1): Node 1 received data
9:1:2.427764850 DEBUG (7): Message sent successfully by node with id: 7 to the parent 1
9:1:2.442123372 DEBUG (1): Node 1 received data
9:1:2.442291218 DEBUG (4): Message sent successfully by node with id: 4 to the parent 1
9:1:2.447708113 DEBUG (1): Node 1 received data
9:1:2.447875959 DEBUG (2): Message sent successfully by node with id: 2 to the parent 1
9:1:2.890136713 DEBUG (0): Node 0 received data
9:1:2.890304558 DEBUG (1): Message sent successfully by node with id: 1 to the parent 0
AVG Aggregation Result: 22
9:1:41.992004387 DEBUG (4): Node 4 received data
9:1:41.992172233 DEBUG (5): Message sent successfully by node with id: 5 to the parent 4
9:1:42.424056986 DEBUG (1): Node 1 received data
9:1:42.424224832 DEBUG (7): Message sent successfully by node with id: 7 to the parent 1
9:1:42.437911971 DEBUG (1): Node 1 received data
9:1:42.438079816 DEBUG (4): Message sent successfully by node with id: 4 to the parent 1
9:1:42.451370201 DEBUG (1): Node 1 received data
9:1:42.451538047 DEBUG (2): Message sent successfully by node with id: 2 to the parent 1
9:1:42.895736656 DEBUG (0): Node 0 received data
9:1:42.895904501 DEBUG (1): Message sent successfully by node with id: 1 to the parent 0
MAX Aggregation Result: 41
```

Παραπάνω φαίνεται ότι μετά την αποστολή των μηνυμάτων από κάθε κόμβο ο πατέρας του λαμβάνει το μήνυμα. Ο λόγος που φαίνονται πρώτα τα μηνύματα λήψης και μετά τα μηνύματα αποστολής οφείλεται στον τρόπο λειτουργίας του event AM.sendDone(). Παρόλα αυτά όλα τα δεδομένα λαμβάνονται σωστά από τους πατεράδες των κόμβων.

Για τον έλεγχο της αποστολής σωστών δεδομενων έχουμε δημιουργήσει ένα καινούργιο κανάλι μέσα στο αρχείο mySimulation.py με όνομα CustomDataChosen.

```
0:14:10.000000010 DEBUG (0): Value generated for nodeID: 0 is: 14
0:14:10.000000010 DEBUG (1): Value generated for nodeID: 1 is: 10
0:14:10.000000010 DEBUG (2): Value generated for nodeID: 2 is: 5
0:14:10.000000010 DEBUG (4): Value generated for nodeID: 4 is: 27
0:14:10.000000010 DEBUG (5): Value generated for nodeID: 5 is: 5
0:14:10.000000010 DEBUG (7): Value generated for nodeID: 7 is: 13
0:14:21.987304708 DEBUG (5): Node ID: 5 - Parent: 4 - sensorVal: 5 - sum: 0 - Children Count 0 - max 0
0:14:22.416992208 DEBUG (7): Node ID: 7 - Parent: 1 - sensorVal: 13 - sum: 0 - Children Count 0 - max 0
0:14:22.431640645 DEBUG (4): Node ID: 4 - Parent: 1 - sensorVal: 27 - sum: 5 - Children Count 1 - max 27
0:14:22.441406270 DEBUG (2): Node ID: 2 - Parent: 1 - sensorVal: 5 - sum: 0 - Children Count 0 - max 0
0:14:22.885742207 DEBUG (1): Node ID: 1 - Parent: 0 - sensorVal: 10 - sum: 50 - Children Count 4 - max 27
0:14:23.330078135 DEBUG (0): Node ID: 0 - Parent: NONE - sensorVal: 14 - sum: 74 - Children count: 6 - max: 27

AVG Aggregation Result: 12

0:14:50.000000010 DEBUG (0): Value generated for nodeID: 0 is: 18
0:14:50.000000010 DEBUG (1): Value generated for nodeID: 1 is: 12
0:14:50.000000010 DEBUG (2): Value generated for nodeID: 2 is: 6
0:14:50.000000010 DEBUG (4): Value generated for nodeID: 4 is: 24
0:14:50.000000010 DEBUG (5): Value generated for nodeID: 5 is: 4
0:14:50.000000010 DEBUG (7): Value generated for nodeID: 7 is: 14
0:15:1.987304708 DEBUG (5): Node ID: 5 - Parent: 4 - sensorVal: 4 - sum: 0 - Children Count 0 - max 0
0:15:2.416992208 DEBUG (7): Node ID: 7 - Parent: 1 - sensorVal: 14 - sum: 0 - Children Count 0 - max 0
0:15:2.431640645 DEBUG (4): Node ID: 4 - Parent: 1 - sensorVal: 24 - sum: 4 - Children Count 1 - max 24
0:15:2.441406270 DEBUG (2): Node ID: 2 - Parent: 1 - sensorVal: 6 - sum: 0 - Children Count 0 - max 0
0:15:2.885742207 DEBUG (1): Node ID: 1 - Parent: 0 - sensorVal: 12 - sum: 48 - Children Count 4 - max 24
0:15:3.330078135 DEBUG (0): Node ID: 0 - Parent: NONE - sensorVal: 18 - sum: 78 - Children count: 6 - max: 24

MAX Aggregation Result: 24
```

Παραπάνω παρατηρούμε τις τιμές που δημιουργούνται για κάθε κόμβο και στη συνέχεια τις τιμές που έχει κάθε κόμβος αμέσως πριν στείλει δεδομένα. Για τον κόμβο 0, που δεν αποστέλλει δεδομένα, παρατηρούμε τα δεδομένα αμέσως πριν κληθεί η συναθροιστική συνάρτηση.

Στο παραπάνω screenshot μας δίνεται και η δυνατότητα να ελέγξουμε ότι το τελικό αποτέλεσμα της συναθροιστικής συνάρτησης είναι επίσης σωστό, είτε για max είτε για average.